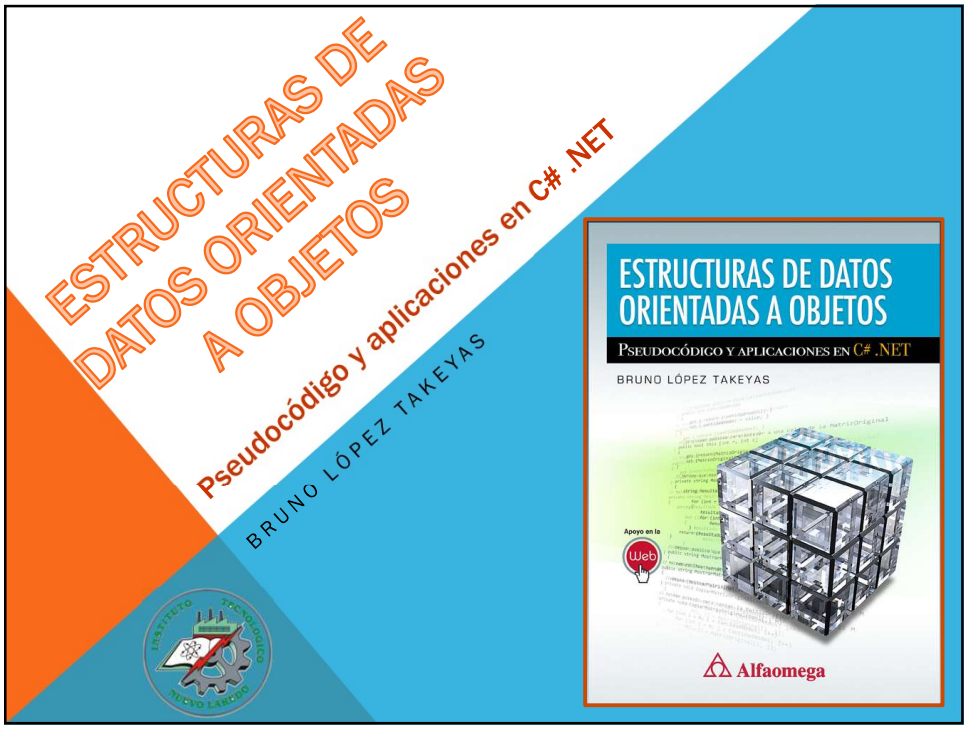
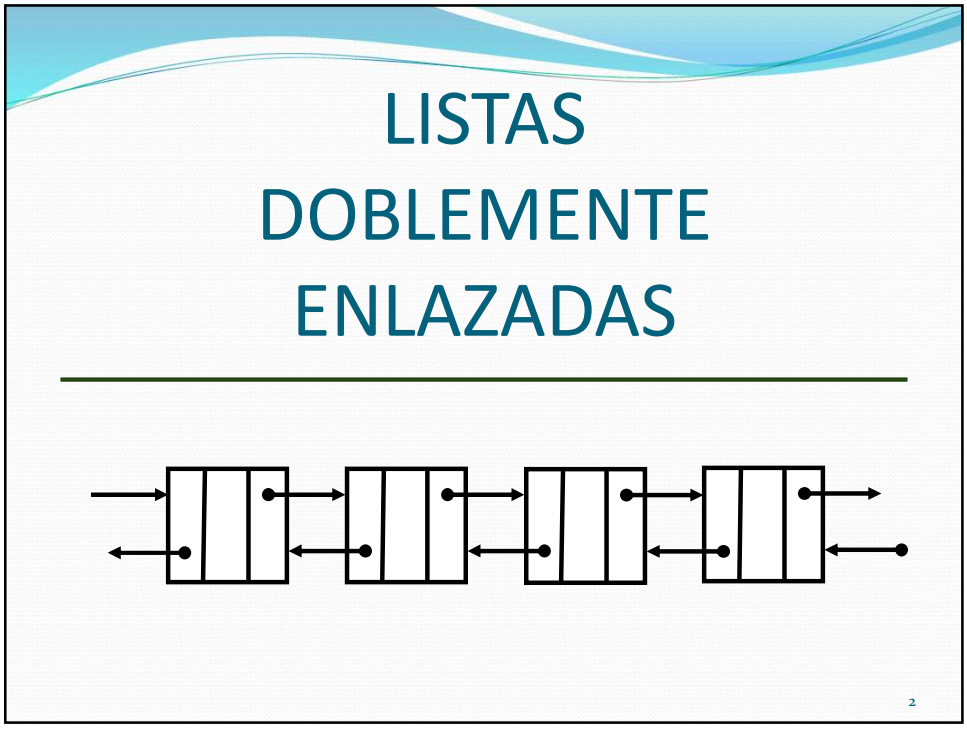

ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
Pseudocódigo y aplicaciones en C# .NET
BRUNO LÓPEZ TAKEYAS
ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
PSEUDOCÓDIGO Y APLICACIONES EN C# .NET
BRUNO LÓPEZ TAKEYAS
Alfaomega


LISTAS DOBLEMENTE ENLAZADAS
2

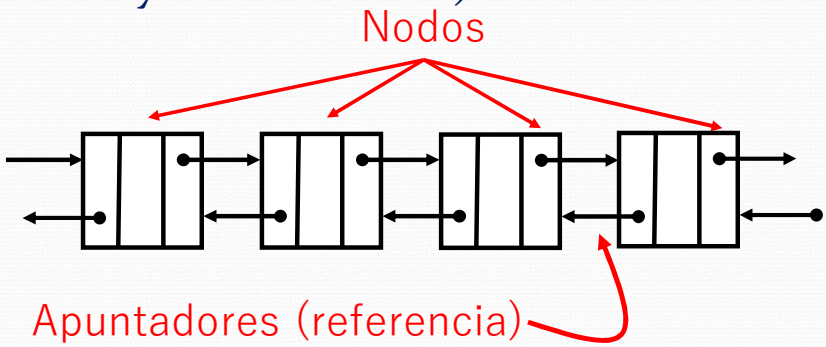
Preguntas detonadoras

- ¿Qué es una lista doble?
- ¿Cómo se clasifican las listas dobles?
- ¿Qué características distinguen a una lista doble?
- ¿Cómo se representa gráficamente una lista doble?
- ¿Qué operaciones se pueden realizar en una lista doble?
- ¿Por qué es una estructura dinámica?
- ¿Cómo se diseña un modelo orientado a objetos con una lista doble?

3

Lista doble

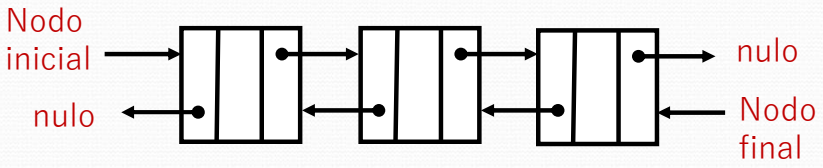
Es una estructura de datos dinámica que se compone de un conjunto de nodos en secuencia enlazados mediante dos apuntadores (uno hacia adelante y otro hacia atrás)



4

Representación gráfica de una lista doble

- *La lista doble tiene un apuntador inicial y un apuntador final*
- *El primero y el último nodos de la lista apuntan a nulo*



5

Tipos de listas dobles

Listas dobles

- *Con datos ordenados*
- *Con datos desordenados*

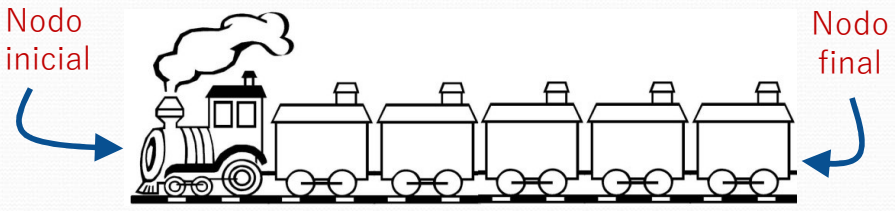
6

Almacenamiento de los datos en una lista doble

- Ordenados:
  - *Se recorren **lógicamente** los nodos de la lista doble*
  - *Se ubica la posición definitiva de un nuevo nodo*
  - *Se mantiene el orden **lógico** de los datos*
- Desordenados
  - *El nuevo dato se agrega al final de la lista doble*

7

Ejemplo de lista doble en la vida cotidiana

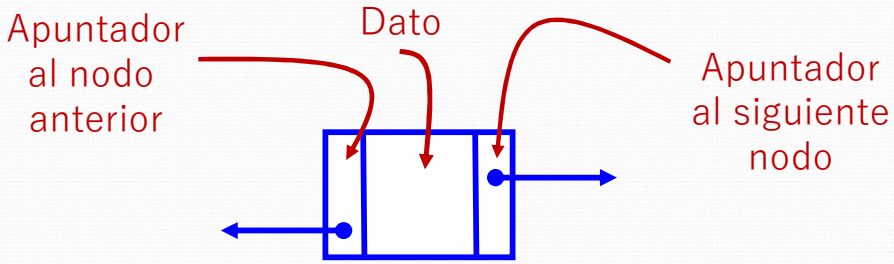


8

Arquitectura de un nodo

Cada nodo tiene 3 secciones:

1. *Dato (puede ser simple o compuesto)*
2. *Apuntador o referencia que enlaza al siguiente nodo en secuencia lógica*
3. *Apuntador que enlaza al nodo anterior*



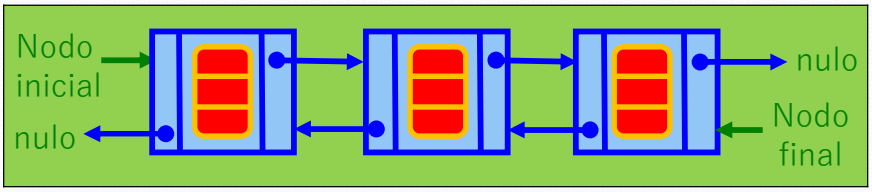
9

Diseño de una lista doble orientada a objetos

- Se identifican 3 tipos de objetos en la lista doble:
  1) *Los objetos con los datos que se desean almacenar y ordenar*
  2) *Los objetos de los nodos*
  3) *El objeto de la lista doble*

10

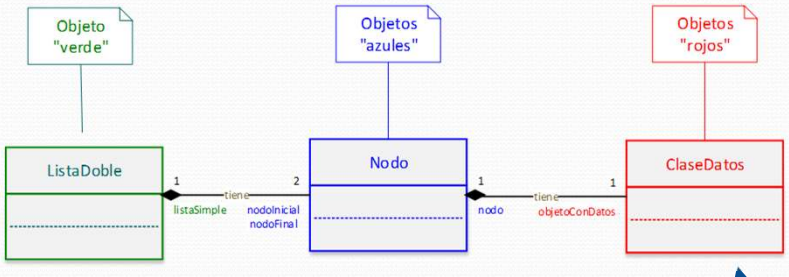
Esquema de los objetos



La **lista doble (objeto verde)** dentro tiene una secuencia lógica de **nodos (objetos azules)** enlazados por apuntadores y cada uno de ellos a su vez tiene dentro un **objeto con los datos** que se desean almacenar y ordenar **(objetos rojos)**

11

Diseño genérico y didáctico de la lista doble



- *Esta clase puede ser...*
- `Empleado`
- `Escuela`
- `Médico`
- *etc. (según lo que se desee almacenar en la lista doble)*

12

Diseño orientado a objetos de una lista doble desordenada

- Se diseña una lista doble con objetos creados por medio de varios tipos de clases:
  - *Clase "roja".- Sirve para crear objetos con los datos que se desean almacenar en la lista*
  - *Clase "azul".- Clase para crear los nodos*
  - *Clase "verde".- Clase para crear el objeto de la lista doble desordenada*

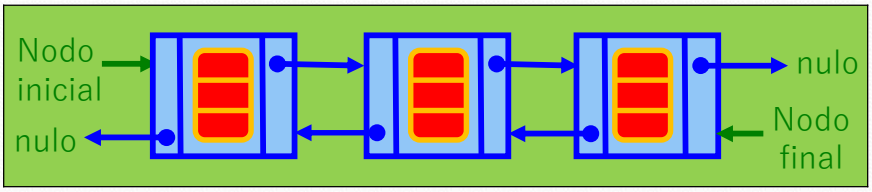
13

Notación de colores

- Los estudiantes deben poner especial atención a los colores usados en las figuras explicativas del resto del curso.
- Los objetos estarán identificados por colores
  - *Objetos "rojos".- Contienen los datos que se desean almacenar y ordenar en la lista*
  - *Objetos "azules".- Representan los nodos de la lista doble*
  - *Objeto "verde".- Es la lista doble*

14

Diseño de clases



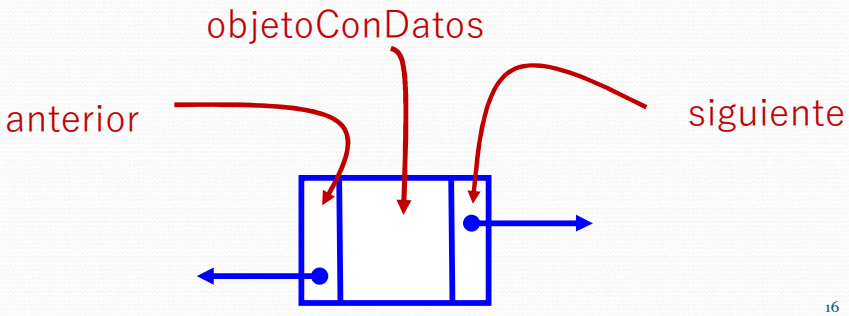
- **Clase "verde"**.- *Define la **composición** entre la lista doble y los nodos que la forman. También tiene los métodos y propiedades para administrar la lista.*
- **Clase "azul"**.- *Define los componentes de los nodos.*
- **Clase "roja"**.- *Definiciones de los datos que se desean almacenar y ordenar en la lista doble.*

15

Declaración del nodo en la lista doble

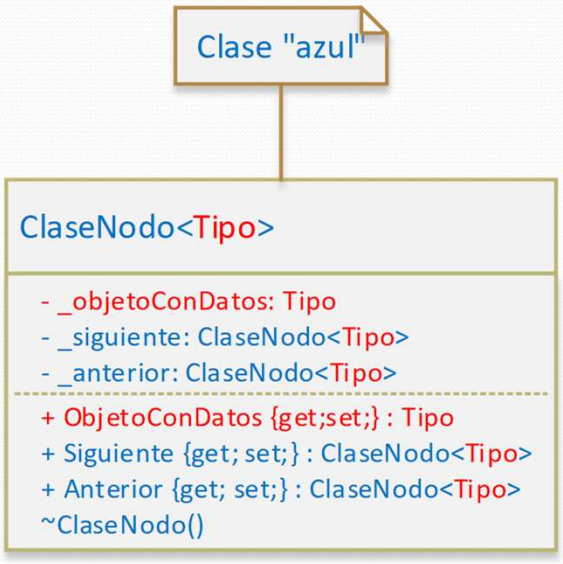
Cada nodo es un obtiene 3 atributos (con sus propiedades):

1. ***objetoConDatos***.- *El nodo recibirá un objeto con los datos que se desean almacenar en la lista doble.*
2. ***siguiente***.- *Apuntador que enlaza al siguiente nodo lógico en la lista doble*
3. ***anterior***.- *Apuntador que enlaza al nodo lógico anterior en la lista doble*



16

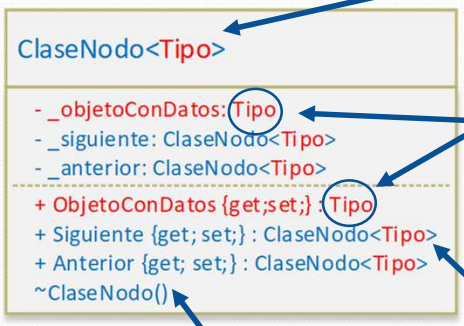
Diseño de la clase de los nodos



17

Diseño de la ClaseNodo

- *Clase parametrizada para recibir cualquier tipo de objeto "rojo"*
- *El parámetro `<Tipo>` define el tipo de objeto "rojo" que estará dentro de cada nodo "azul"*
- *Los apuntadores `Siguiente` y `Anterior` NO almacenan otro nodo "azul" sino apuntan hacia otro nodo de su mismo tipo*

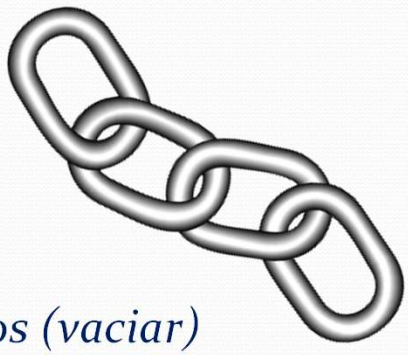


- *Al eliminar un nodo "azul", su destructor elimina el objeto "rojo" que contiene*

18

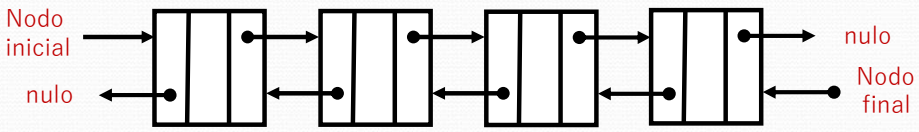
Operaciones en una lista doble

- *Creación de la lista*
- *Inserción de un dato*
- *Eliminación de un dato*
- *Recorridos (en ambos sentidos)*
- *Búsqueda*
- *Eliminar todos los nodos (vaciar)*



19

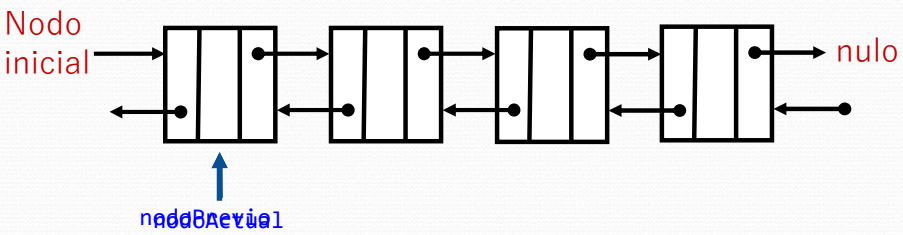
Recorrido de los nodos en una lista doble



- *Se verifica que la lista no esté vacía*
- *El recorrido empieza en el* `NodoInicial` *o en el* `NodoFinal`
- *Si se empieza por el* `NodoInicial` *entonces se avanza al próximo nodo a través del apuntador* `Siguiente`
- *Si se empieza por el* `NodoFinal` *entonces se retrocede al nodo anterior a través del apuntador* `Anterior`
- *En algunos casos es necesario guardar en una variable el nodo previo al cambiar de nodo*
- *El recorrido termina al llegar al nodo que apunta a nulo*

20

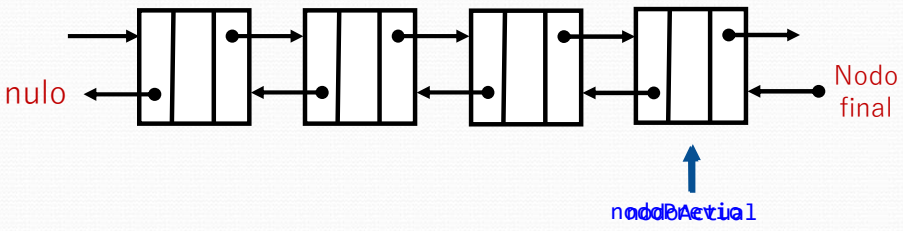
Ejemplo de recorrido hacia adelante



Termina el recorrido

21

Ejemplo de recorrido hacia atrás

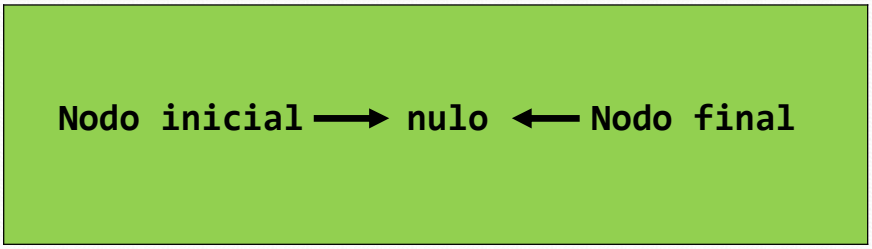


Termina el recorrido

22

Creación de una lista doble desordenada

Cuando se crea una lista doble tanto el nodo inicial como nodo final el apuntan a nulo



23

Situaciones críticas

- *Son las situaciones que se pueden presentar al realizar operaciones con estructuras de datos*
- *El programador debe prever para diseñar algoritmos eficientes*

24

Inserción de datos en una lista doble desordenada

- Situaciones críticas:
  - *Alta a lista doble vacía*
  - *Alta al final.- Se agrega el dato al final de la lista doble*
  - ***NO** permitir datos duplicados*

Se debe recorrer la lista doble completa antes de insertar un nodo nuevo para validar que no exista duplicado

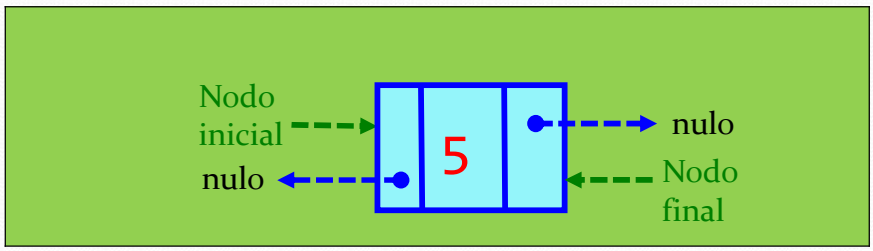
Recuerde:
La lista doble almacena los datos desordenados

25

Alta a una lista doble desordenada vacía

Cuando se detecta la lista doble vacía:

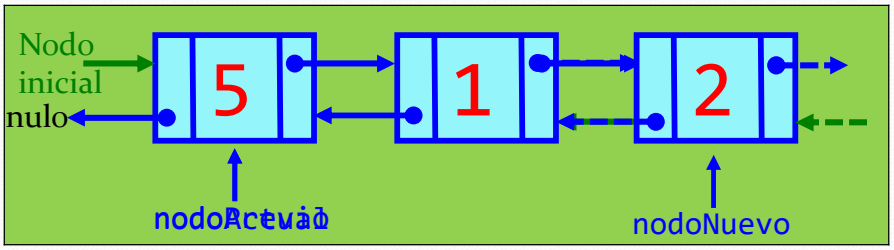
1) *Crear el nuevo nodo "azul"*
2) *Insertar el dato "rojo"*
3) *El nuevo nodo apunta a nulo en el siguiente y en el anterior*
4) *El nodo inicial apunta al nuevo nodo*
5) *El nodo final apunta al nuevo nodo*



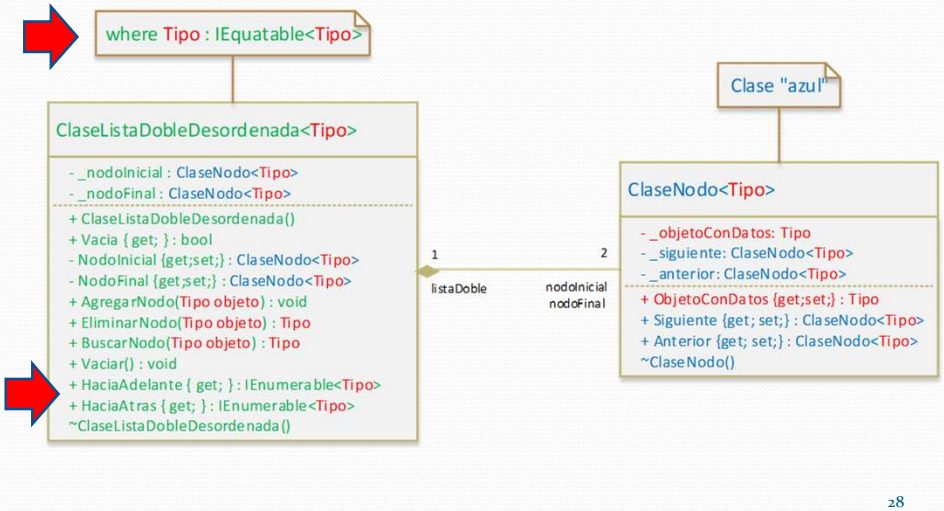
Alta al final en una lista doble desordenada

Cuando se inserta un dato al final:

1) *Recorrer la lista para verificar duplicado*
2) *Crear el nodo nuevo "azul"*
3) *Insertar el dato "rojo"*
4) *El nodo previo apunta al nodo nuevo con el siguiente*
5) *El nodo nuevo apunta al nodo previo con el anterior*
6) *El nodo nuevo apunta a nulo con el siguiente*
7) *El nodo final apunta al nodo nuevo*



Diseño de la clase de la lista doble desordenada



28

Composición Lista-Nodo

- *¿Por qué es una composición **1..2** si la lista doble tiene muchos nodos dentro de ella?*
  - *Porque la cardinalidad de una composición se define por la cantidad de atributos de tipo "parte" contenidos en la clase del "todo" (regla 1 de la composición)*

29

ClaseListaDobleDesordenada<Tipo>

- *Clase parametrizada para prepararla para que pueda recibir cualquier `Tipo` de objeto "rojo"*
- *Tiene una restricción de tipos para "obligar" a que la clase "roja" implemente `IEquatable`*
  - *Se requiere el método `Equals()` para buscar un objeto "rojo" almacenado en la lista*
- *¿Por qué no tiene restricción de tipos para `IComparable`?*

30

Componentes de la clase

- `_nodoInicial`.- *Atributo privado que apunta al primer nodo de la lista doble*
- `_nodoFinal`.- *Atributo privado que apunta al último nodo de la lista doble*
- `ClaseListaDobleDesordenada()`.- *Es el constructor que inicializa vacía la lista.*
- `Vacia`.- *Propiedad pública booleana de solo lectura para detectar si la lista está vacía (devuelve `true` si la lista está vacía).*

31

Componentes de la clase *(cont.)*

- `NodoInicial`.- *Propiedad privada que apunta al primer nodo de la lista doble*
- `NodoFinal`.- *Propiedad privada que apunta al último nodo de la lista doble*
- `AgregarNodo(Tipo objeto):void` .- *Método público que recibe como parámetro el objeto "rojo" que se desea almacenar en la lista doble.*
- `EliminarNodo(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" que se desea borrar de la lista doble.* ***Devuelve*** *el objeto "rojo" eliminado.*

32

Componentes de la clase *(cont.)*

- `BuscarNodo(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" que se desea consultar en la lista. Devuelve el objeto "rojo" localizado.*

- `Vaciar():void` .- *Método público que recorre la lista doble para eliminar todos los nodos "azules" con sus respectivos objetos "rojos"*

33

Componentes de la clase *(cont.)*

- `~ClaseListaDobleDesordenada()`.- *Destructor que invoca al método* `Vaciar()` *para eliminar todos los* `nodos` *de la lista doble.*

34

Iteradores de la lista doble

- *Debido a que la lista doble se recorre en ambos sentidos entonces se requieren 2 iteradores.*
- *Se implementan a través de propiedades de solo lectura*
  - `HaciaAdelante {get;}:IEnumerable<Tipo>`.- *Iterador que recorre la lista doble partiendo del* `NodoInicial` *para visitar cada nodo "azul" y consultar su objeto "rojo"*
  - `HaciaAtras {get;}:IEnumerable<Tipo>`.- *Iterador que recorre la lista doble partiendo del* `NodoFinal` *para visitar cada nodo "azul" y consultar su objeto "rojo"*

35

Tarea 1.08.- DF Listas dobles (constructor, vacía e iteradores)

- Subir a MS Teams el archivo *.vsdx (Microsoft Visio) con diagrama de flujo de:
  - *Constructor de la lista doble*
  - *Propiedad para detectar si la lista doble está vacía*
  - *Iteradores*
    - *HaciaAdelante*
    - *HaciaAtras*
  - *Método*
    - *Buscar*

36

Diseño de la clase "roja"

- *Requisitos: Debe contener al menos un dato de los siguientes tipos:*
  - *Int*
  - *Double*
  - *String*
  - *Char*
  - *DateTime*
  - *Bool*
  - *String con la ruta del archivo que contiene una fotografía del objeto*

Se recomienda consultar las filminas
"El lenguaje C# y diseño de formas"
Para usar el PictureBox

37

Diseño de la clase "roja" (*cont.*)

IMPORTANTE

Considere un dato dentro de su clase "roja" que sea único e irrepetible que sirva para identificar a un objeto y que sea el criterio de comparación entre objetos

(Una clave para identificar y comparar objetos "rojos")

38

Tarea 1.09

- *Resolver la Tarea 1.09.- Diseño de la clase "roja" de la lista doble en MS Teams*
- *Subir el archivo *.vsdx con el diagrama de la clase "roja" elaborado en Microsoft Visio*
- *Incluir las interfaces correspondientes*

39

Tarea 1.10.- DF Listas dobles (Agregar)

- Subir a MS Teams el archivo *.vsdx (Microsoft Visio) con diagrama de flujo de:
  - *Método para agregar un objeto "rojo" a la lista doble*

40

Diseño de la forma de la aplicación visual

- *Requisitos: Debe contener al menos uno de estos controles visuales:*
  - *Textbox*
  - *Button*
  - *ComboBox*
  - *DateTimePicker*
  - *CheckBox*
  - *PictureBox*
  - *DataGridView*
  - *RadioButton*

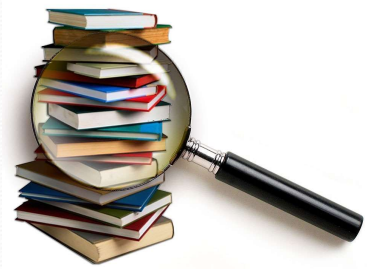
Elija el control adecuado para cada dato capturado

Se recomienda consultar las filminas ***"Uso de los controles visuales"***

41

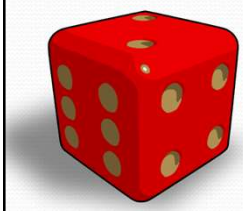
Investigación

- *Agregue un botón a su aplicación para crear* `10` *nodos con datos generados de manera aleatoria*
- *Muestre los datos generados en el dataGridView*



42

LECTURA



Para generar datos aleatorios, se recomienda la lectura de los apuntes

¿Cómo generar datos aleatorios en C#?

Incluye generar:

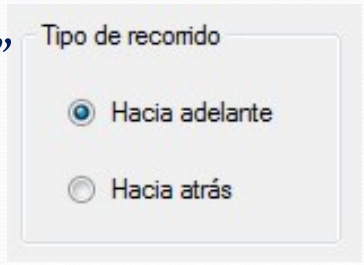
- *Nombres aleatorios*
- *Sexo*
- *Fecha*
- *Datos booleanos*
- *etc.*

https://nlaredo.tecnm.mx/takeyas/Apuntes/Fundamentos%20de%20Programacion/Apuntes/07.-Aleatorios.pdf

43

Uso de los iteradores

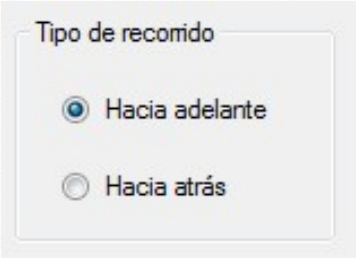
- *Agregue radioButtons a su aplicación para seleccionar el tipo de recorrido en la lista doble*
- *Muestre los datos de los objetos "rojos" en el dataGridView dependiendo de la selección*



44

Uso de los iteradores (*cont.*)

Automáticamente *se deben actualizar los datos del dataGridView al seleccionar el tipo de recorrido haciendo "click" en alguno de los radioButtons*



45

¿Cómo invocar los iteradores de la lista doble?

Los iteradores se implementan a través de propiedades de solo lectura:

- `HaciaAdelante { get;} : IEnumerable<Tipo>`
- `HaciaAtras    { get;} : IEnumerable<Tipo>`

- *Ejemplos:*

```
foreach(Empleado miEmpleado in miListaDoble.HaciaAdelante)
```

```
foreach(Empleado miEmpleado in miListaDoble.HaciaAtras)
```

46

Tarea 1.11.- Diseño de la forma de listas dobles

- Diseñar la forma en C#:
- *Capturar los datos usando los controles visuales adecuados*
- *Agregar un dataGridView de solo lectura para visualizar los datos de los objetos "rojos"*
- *Implementar el método para agregar objetos "rojos" a la lista doble y visualizarlos en el dataGridView (de acuerdo al tipo de recorrido seleccionado)*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

47

Tarea 1.12.- DF Listas dobles (Buscar)

- Subir a MS Teams el archivo *.vsdx (Microsoft Visio) con diagrama de flujo de:
  - *Método para buscar un objeto "rojo" en la lista doble (debe devolver el objeto encontrado)*

48

Eliminación de datos en una lista doble desordenada

- *Se recorre la lista doble (en cualquier sentido) para localizar el nodo con el objeto "rojo" que se desea eliminar*
- *Durante el recorrido es necesario "guardar" el nodo previo*
- *No se puede interrumpir la búsqueda por anticipado ya que la lista doble almacena datos desordenados*

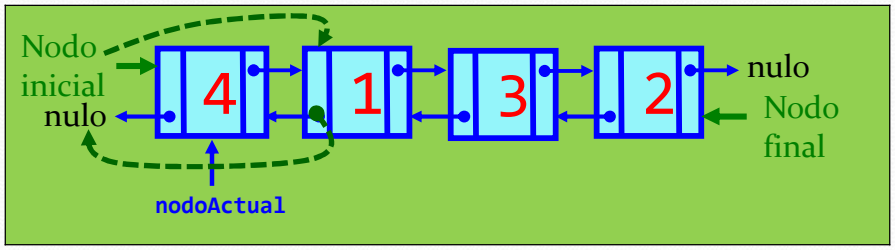
Situaciones críticas de las bajas en una lista doble desordenada

- *Baja a lista vacía.- Se dispara una excepción*
- *Baja al principio.- Se elimina el dato apuntado por el nodo inicial*
- *Baja intermedia.- Se elimina un dato ubicado entre otros*
- *Baja al final.- Se elimina el último dato de la lista doble*
- *Baja del único nodo*
- *Verificar la existencia del dato*

Baja al principio en una lista doble desordenada

Cuando se elimina el primer nodo de la lista doble:

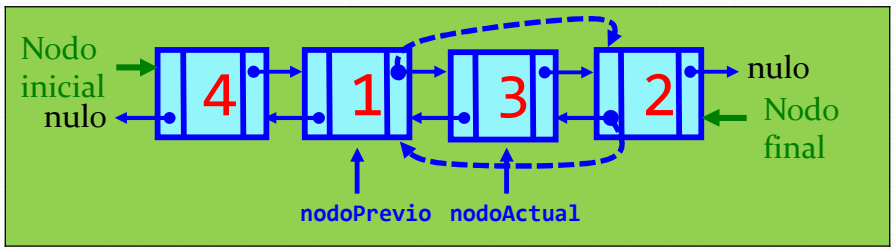
1) *El NodoInicial apunta al siguiente del nodo actual*
2) *El nodo siguiente del nodo actual apunta a nulo con su anterior*
3) *Se elimina el nodoActual*



Baja intermedia en una lista doble desordenada

Cuando se elimina un nodo intermedio de la lista doble:

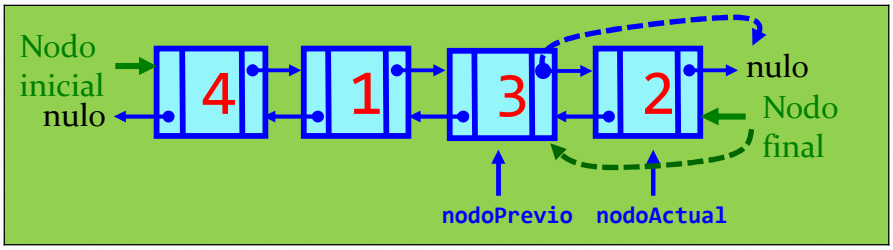
1) *El siguiente del nodo previo apunta al siguiente del nodo actual*
2) *El nodo siguiente del nodo actual apunta al nodo previo con su anterior*
3) *Se elimina el nodoActual*



Baja al final en una lista doble desordenada

Cuando se elimina el último nodo de la lista doble:

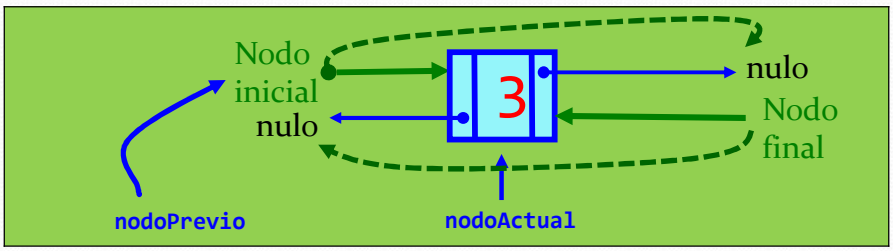
1) *El siguiente del nodo previo apunta a nulo*
2) *El nodo final apunta al nodo previo*
3) *Se elimina el nodoActual*



Baja del único nodo de una lista doble desordenada

Cuando se elimina el único nodo de la lista doble:

1) *El nodo inicial apunta a nulo*
2) *El nodo final apunta a nulo*
3) *Se elimina el nodoActual*



Eliminación de objetos

- *La forma natural de borrar un objeto es asignarle el valor null*

```
// Creación del objeto "azul" del nodoActual
ClaseNodo<Tipo> nodoActual = new ClaseNodo<Tipo>();
   .
   .
nodoActual = null; // Eliminación del nodoActual
```

- *Sin embargo, no todos los datos aceptan el valor null, entonces... ¿cómo se eliminarían?*

55

Destructor de la clase "azul"

- *La* `ClaseNodo<Tipo>` *es parametrizada y está preparada para recibir un objeto "rojo" de cualquier tipo.*
- *El destructor de la clase "azul" elimina el objeto con datos "rojo" que contiene.*
- *Utiliza* `default(Tipo)` *para eliminar el objeto "rojo" porque desconoce si el ObjetoConDatos acepta el valor null.*

```
~ClaseNodo() // Destructor de la clase "azul"
{
     // Elimina el ObjetoConDatos "rojo"
     ObjetoConDatos = default(Tipo);
}
```

Tarea 1.13.- DF Listas dobles (Eliminar)

- Subir a MS Teams el archivo *.vsdx (Microsoft Visio) con diagrama de flujo de:
  - Método para eliminar un objeto "rojo" de la lista
    - ***Devolver*** *el objeto "rojo" eliminado*

57

Operación del método para eliminar un nodo de la lista

- *Seleccionar un renglón del dataGridView y después oprimir el botón "Eliminar"*
- *Al seleccionar un renglón del dataGridView se deben mostrar los datos en los controles adecuados*
- *Confirmar la operación*
- ***Devolver*** *el objeto "rojo" eliminado*

58

Vaciar la lista doble desordenada

- Situaciones críticas:
  - *Verificar si la lista doble ya está vacía*
  - *Recorrer los nodos "azules" de la lista*
  - *Durante el recorrido, guardar el* `nodoPrevio`
  - *Eliminar el* `nodoPrevio`
  - *Al terminar, el* `NodoInicial` *y el* `NodoFinal` *deben apuntar a nulo*

59

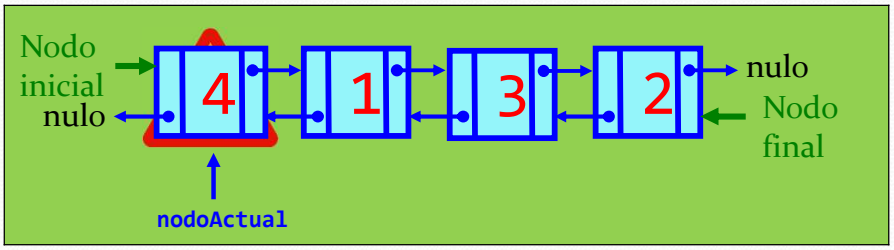
Precaución al vaciar la lista doble

- **NOTA IMPORTANTE:**
  - *Durante el recorrido de los nodos "azules" de la lista doble,* ***NO*** *se debe eliminar el* `nodoActual`
  - *Si se eliminara el* `nodoActual` *se pierde el apuntador* `Siguiente`
  - *Se perdería la secuencia lógica de los nodos "azules"*

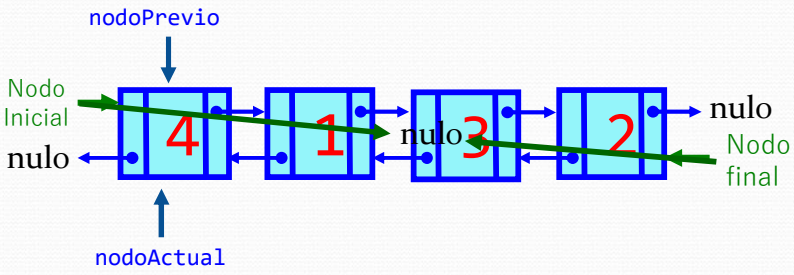
60

Precaución al vaciar la lista doble

- NOTA IMPORTANTE:
  - *Si se eliminara el* `nodoActual` *. . .*
  - *¿Cómo verificaría cuál es el siguiente nodo?*
  - *Por lo tanto* ***NO*** *debe eliminarse el* `nodoActual` *sino el* `nodoPrevio`



Ejemplo: Vaciar la lista doble



Termina el recorrido

62

Tarea 1.14.- DF Listas dobles (Vaciar y Destructor)

- Subir a MS Teams los archivos *.vsdx (Microsoft Visio) con diagramas de flujo de:
  - *Método para vaciar la lista doble*
  - *Destructor de la lista doble*

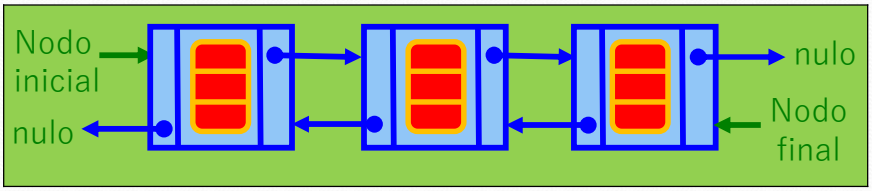
63

Tarea 1.15.- Aplicación completa de listas dobles

- *Agregar un botón "Vaciar"*
- *Solicitar al usuario que confirme las operaciones (insertar, eliminar, vaciar, etc.). Preguntarle si está seguro que desea realizar la operación solicitada*
- *Mostrar los mensajes adecuados*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

64

Nivel de abstracción



- *No se debe perder de vista que los objetos "rojos" son privados, por lo tanto sus componentes son inacesibles para el objeto "verde"*

65

Nivel de abstracción (*cont.*)

- *El objeto "verde" recibe un objeto "rojo" y lo compara para almacenarlo ....*

  ¡¡¡ SIN SABER LO QUE TIENE DENTRO !!!

- *¿Cómo es posible que el objeto "verde" compare y almacene objetos "rojos" sin tener acceso a sus componentes?*

66

"Pensar en objetos …"

- *El objeto "verde" **NO** compara los objetos "rojos" (ellos mismos se comparan entre sí).*
- *El objeto "verde" **NO** requiere acceso a los componentes de los objetos "rojos" para manipularlos.*
- *Recibe cualquier tipo de objetos "rojos" …*

¡¡¡ SIN MODIFICAR NI UNA LÍNEA DE SU CÓDIGO!!!

- ***¿Cómo lo logra? …***
  - *Clases parametrizadas*
  - *Uso de interfaces*
  - *Composición*
  - *Comportamiento polimórfico*
  - *Restricción de tipos*

67

Diseño de una clase polimórfica

- La clase de la lista tiene estas características:
  - *Almacena objetos "rojos" desordenados*
  - *No permite duplicados*

68

Diseño de una clase polimórfica *(cont.)*

- *¿Qué cambios se harían para que la misma clase "**verde**" almacene los datos ordenados?*
- *¿Qué cambios se harían para que la misma clase "**verde**" permita objetos "rojos" duplicados?*

69

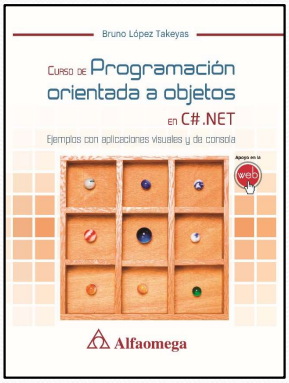
Diseño de una clase polimórfica *(cont.)*

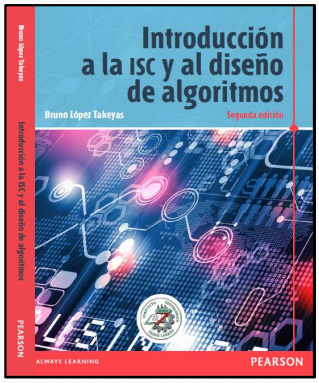
- *¿Qué cambios se harían para que la misma clase "**verde**" muestre cualquiera de los siguientes comportamientos:*
- *Objetos "rojos" ordenados sin duplicados*
- *Objetos "rojos" ordenados con duplicados*
- *Objetos "rojos" desordenados sin duplicados*
- *Objetos "rojos" desordenados con duplicados?*

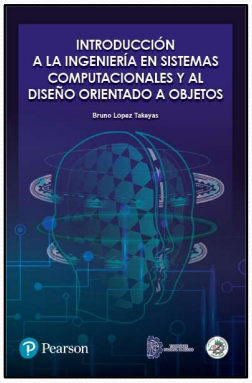
70

Otros libros del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas